Admissible Substitutions in the Calculus of Natural Deduction

Anton Freund

Seminar talk at the Institute of Mathematics, LMU Munich

June 15, 2011

Content of my talk:

- Short motivation: What are substitutions and what are they for?
- Substitutions in object terms: A very detailed analysis of occuring problems and possible solutions.
- Implementation of substitutions in Minlog.
- Outlook: Substitutions in constants (assumption constants, recursion operator) and inductively defined predicates. Substitution and proof extraction.

- Substitution: replacement of variables by appropriate terms.
- Substitutions as *simultaneous substitutions*:

Definition

A substitution is a list of pairs $\vartheta := ((X_1\ Y_1), \ldots, (X_n\ Y_n))$, such that the X_i are pairwise distinct and for each $i \in \{1, \ldots, n\}$ the following folds:

1. X_i is a type, object, predicate or assumtion variable and Y_i is a type, object, comprehension or proof term.
2. $X_i \neq Y_i$.

- Situations in which we need to substitute: Specialization in proofs, β-conversion.

Deeper account of the use of substitutions:

- Quantification only over *object* variables.
- As a consequence: induction on natural numbers not as an *axiom* but only as an *axiom scheme*:

$$X(0) \to \forall_n(X(n) \to X(Sn)) \to \forall_n X(n).$$

- Solution: Let A be an axiom scheme, ϑ an (admissible) substitution and $A\vartheta$ an application of ϑ to A. Then $A\vartheta$ counts as an axiom.
  For the axiom scheme above and a suitable comprehension term π the application of $\vartheta := ((X\,\pi))$ yields the axiom

$$\pi(0) \to \forall_n(\pi(n) \to \pi(Sn)) \to \forall_n \pi(n).$$

- Similarly: substitution for the parameters of inductively defined predicates.

- For the moment consider very simple object terms only, corresponding to typed lambda-calculus.
- In particular no account for inner structure of base types (e.g. free algebras as base types), no recursion operator included.

Definition (Object terms)

For every type our language includes countably many object variables of that type. Object terms are built out of those inductively:

1. Any object variable of type σ is an object term of type σ.
2. If x is an object variable of type σ and r is an object term of type τ then $\lambda_x r$ is an object term of type $\sigma \to \tau$.
3. If r is an object term of type $\sigma \to \tau$ and s is an object term of type σ, then rs is an object term of type τ.
   It is obvious how to define the set $\mathrm{FV}_o(r)$ of free variables.

- Substitutions have to be compatible with the type structure of our language.
- Consider the following: Let $\overline{r}$ denote the type of an object term r. We would like to define the application of a substitution ϑ to $r^{\sigma\to\tau}s^{\tau}$ to be

$$(rs)\vartheta := (r\vartheta)(s\vartheta).$$

  But for $(r\vartheta)(s\vartheta)$ to be an object term, $\overline{r\vartheta}$ has to be of the form $\rho_1 \to \rho_2$ where $\rho_1 = \overline{s\vartheta}$.
- Solution: If we could guarantee that for every object term r we have $\overline{r\vartheta} = \overline{r}\vartheta$, then the above would hold since

$$\overline{r\vartheta} = \overline{r}\vartheta = (\sigma \to \tau)\vartheta = \sigma\vartheta \to \tau\vartheta = \overline{s\vartheta} \to \tau\vartheta,$$

  where substitution in types is defined inductively in the obvious way (all types are built from type variables by arrow type formation in the present context).

- How to guarantee that $\overline{r\vartheta} = \overline{r}\vartheta$ holds?
- First demand this for object variables. Following a suggestion by Prof. Wilfried Buchholz we define:

Definition (Admissible substitutions)

A substitution ϑ is called admissible for an object variable x if we have $\overline{x\vartheta} = \overline{x}\vartheta$. Furthermore ϑ is called admissible for an objectterm r if ϑ is admissible for all $x \in \mathrm{FV}_o(r)$.

- Now define the application $r\vartheta$ by induction on r.
- Show inductively that $r\vartheta$ is an object term s.t. $\overline{r\vartheta} = \overline{r}\vartheta$.

- Another problem with substitutions: Renaming of bound variables necessary to avoid capture.
- For $r = \lambda_x y$ and $\vartheta = ((y\,x))$ we should *not* have $r\vartheta = \lambda_x x$ but rather $r\vartheta = \lambda_z x$ for some new variable z.
- If we don't fix what new variable to choose, then we can't define *the* result of the application of ϑ to r.

- First solution: Enumerate the variables and choose the first suitable variable from the list.
- Advantage of this solution: This will be the way one implements substitution (e.g. in Minlog).
- Disadvantages:
  - Rather ad hoc: If it doesn't matter which variable we choose then this fact should somehow be reflected by our theory.
  - Consider the following: We ultimately want to define the application of substitutions to proof terms. If M proves the formula A, then $M\vartheta$ should prove $A\vartheta$. Now let M be a proof of a formula $\forall_x(x = x) \to A$ and let N be a proof of $\forall_x(x = x)$. Then MN is a proof of A. We would like to assure that $(MN)\vartheta = (M\vartheta)(N\vartheta)$ is a proof of $A\vartheta$. But for this, the antecedent of $(\forall_x(x = x) \to A)\vartheta$ should be the same formula as $(\forall_x(x = x))\vartheta$. But who guarantees that after possible renamings the bound variable is still the same in both cases? In fact one can give examples where this isn't the case.

For the given reasons we will choose a solution:

- We will define *Alpha-equality* on object terms. $r =_\alpha s$ is supposed to mean that r and s are equal modulo the renaming of bound variables.
- We will *not* define substitution as an operation on object terms. Rather we will define a relation between object terms with the intuitive meaning that s is *one* result of the application of ϑ to r.
- In a second step we will show the following: If $r =_\alpha s$ are object terms and ϑ is a substitution, then we have $r\vartheta =_\alpha s\vartheta$ whenever $r\vartheta$ and $s\vartheta$ are *arbitrary* applications of ϑ to r and s. In this sense we can view substitution as an *operation* on object terms *modulo Alpha-equality*.

We now carry out the described steps in detail.

- Consider very basic object terms only: typed lambda calculus.
- Call lambda terms "object terms" as opposed to formulas, comprehension terms and proof terms, which we will consider later.
- In particular don't consider inner structure of base types (e.g. as free algebras). No recursion operator for term formation.

Definition (Types)

Types are built from type variables by arrow type formation $\sigma \to \tau$.
Symbols used for type variables and types: σ, τ.

Definition (Object terms)

For each type our language includes countably many object variables of this type. Object terms and their free variables are defined by:

1. x^σ, $\mathrm{FV}_\mathrm{o}(x) := \{x\}$,
2. $(\lambda_{x^\sigma} r^\tau)^{\sigma\to\tau}$, $\mathrm{FV}_\mathrm{o}(\lambda_x r) := \mathrm{FV}_\mathrm{o}(r)\backslash\{x\}$,
3. $(r^{\sigma\to\tau} s^\sigma)^\tau$, $\mathrm{FV}_\mathrm{o}(rs) := \mathrm{FV}_\mathrm{o}(r) \cup \mathrm{FV}_\mathrm{o}(s)$.

Let $\overline{r}$ denote the type of r.
Symbols for object variables: x, y, z. For object terms: r, s, t.

Definition (Substitution in types)

Substitution in types is inductively defined by:

1. $\sigma\vartheta := \begin{cases} \vartheta(\sigma) & \text{if } \sigma \in \mathrm{dom}(\vartheta), \\ \sigma & \text{otherwise,} \end{cases}$
2. $(\sigma \to \tau)\vartheta := (\sigma\vartheta) \to (\tau\vartheta)$.

Definition (Admissible substitutions)

A Substitution ϑ is called admissible for an object variable x if $\overline{x\vartheta} = \overline{x}\vartheta$, where

$$x\vartheta := \begin{cases} \vartheta(x) & \text{if } x \in \mathrm{dom}(\vartheta), \\ x & \text{otherwise.} \end{cases}$$

Furthermore ϑ is called admissible for an object term r if ϑ is admissible for all $x \in \mathrm{FV}_o(r)$.

Definition

Let ϑ be admissible for r. We call s an application of ϑ to r if one of the following holds:

1. $r = x$ for an object variable x and $s = x\vartheta$.
2. $r = \lambda_x r'$, $s = \lambda_y s'$ and the following holds: (i) y is an object variable fulfilling $\overline{y} = \overline{x}\vartheta$ and $y \notin \bigcup_{z \in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta)$ and (ii) s' is an application of ϑ^y_x to r'.
3. $r = r_1 r_2$, $s = s_1 s_2$ and s_i is an application of ϑ to r_i for $i = 1, 2$.

Theorem

Let ϑ be admissible for r. Then the following holds:

1. *There is an object term s such that s is an application of ϑ to r.*
2. *If $r\vartheta$ is any application of ϑ to r then we have $\overline{r\vartheta} = \overline{r}\vartheta$ and $FV_o(r\vartheta) = \bigcup_{x \in FV_o(r)} FV_o(x\vartheta)$.*

Proof: We show the claims by simultaneous induction on r.
Case 1. $r = x$. By definition of a substitution $x\vartheta$ is an object term. Thus $s := x\vartheta$ satisfies (1). Furthermore any application of ϑ to x is equal to $x\vartheta$. As ϑ is admissible for x we have $\overline{x\vartheta} = \overline{x}\vartheta$. Since $\mathrm{FV_o}(r) = \{x\}$ we also have $\mathrm{FV_o}(x\vartheta) = \bigcup_{x \in \mathrm{FV_o}(r)} \mathrm{FV_o}(x\vartheta)$.
Case 2. $r = \lambda_x r'$. Choose a object variable y such that $\overline{y} = \overline{x}\vartheta$ and $y \notin \bigcup_{z \in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta)$. Then ϑ^y_x is admissible for r' (requires

proof!). By induction hypothesis there is a term s' which is an application of ϑ^y_x to r'. Then $s := \lambda_y s'$ is as required. Furthermore any application $r\vartheta$ is of the form $\lambda_y(r'\vartheta^y_x)$ where $\overline{y} = \overline{x}\vartheta$, $y \notin \bigcup_{z\in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta)$ and where $r'\vartheta^y_x$ is an application of ϑ^y_x to r'. Using the induction hypothesis we have $\overline{r\vartheta} = \overline{y} \to \overline{r'\vartheta^y_x} = \overline{x}\vartheta \to \overline{r}'\vartheta^y_x$. Since ϑ and ϑ^y_x coincide on all type variables we have $\overline{r}'\vartheta^y_x = \overline{r}'\vartheta$ (requires proof!). Using this, we have $\overline{r\vartheta} = \overline{x}\vartheta \to \overline{r}'\vartheta = \overline{r}\vartheta$.Using the induction hypotheseis and $\mathrm{FV_o}(r) = \mathrm{FV_o}(r')\backslash\{x\}$ yields $\mathrm{FV_o}(r\vartheta) = \mathrm{FV_o}(r'\vartheta^y_x)\backslash\{y\} =$ $= [\bigcup_{z\in \mathrm{FV_o}(r')} \mathrm{FV_o}(z\vartheta^y_x)]\backslash\{y\} = [\bigcup_{z\in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta)]\backslash\{y\}$ and the claim follows with the variable condition $y \notin \bigcup_{z\in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta)$.

Case 3. $r^\tau = r_1^{\sigma\to\tau} r_2^\sigma$. For $i = 1, 2$ we have $\mathrm{FV_o}(r_i) \subseteq \mathrm{FV_o}(r)$, thus ϑ is admissible for r_1 and r_2. By the induction hypothesis there exist object terms s_1, s_2 which are applications of ϑ to r_1, r_2 and which satisfy $\overline{s_1} = \overline{r_1}\vartheta = (\sigma \to \tau)\vartheta = \sigma\vartheta \to \tau\vartheta$ and

$\overline{s_2} = \overline{r_2}\vartheta = \sigma\vartheta$. Thus $s := s_1 s_2$ is an object term satisfying (1). The further claims follow easily with the induction hypothesis. □

- The fact that $\mathrm{FV_o}(r\vartheta) = \bigcup_{x \in \mathrm{FV_o}(r)} \mathrm{FV_o}(x\vartheta)$ shows that the condition $y \notin \bigcup_{z \in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta)$ for a new variable in the abstraction case is sufficient and necessary.
- Goal: Use this to check the implementation of `term-substitute` in `term.scm`.

```
 1 (define (term-substitute term tosubst)
 2 [...]
 3 (let* ((var (term-in-abst-form-to-var term))
 4       (kernel (term-in-abst-form-to-kernel term))
 5       (type (var-to-type var))
 6       (tovars (map car tosubst))
 7       (active-vars (intersection tovars (term-to-free term)))
 8       (active-subst (list-transform-positive tosubst
 9         (lambda (x) (member (car x) active-vars))))
10       (active-terms (map cadr active-subst))
11       (new-var
12         (if ;type is not changed
13  (null? (intersection (type-to-free type) (map car tsubst)))
14  (if ;there is no clash
15          (and (not (member var tovars))
16               (not (member var (apply union (map term-to-free active-terms)))))
17       var
18       (var-to-new-var var))
19    (type-to-new-var (type-substitute type tsubst))))
20       (new-subst (if (equal? var new-var)
21        active-subst
22        (cons (list var (make-term-in-var-form new-var))
23      active-subst))))
24   (make-term-in-abst-form
25    new-var (term-substitute kernel (append tsubst new-subst)))))
```

To avoid superfluous renaming one can change lines 15 and 16 to

```
(not (member var (apply union (map term-to-free active-terms))))
```

Plan for the following:

- Define Alpha-equality on object terms (definition by Robert Stärk, as implemented in Minlog).
- $r =_\alpha s$ is supposed to mean that r and s are equal modulo renaming of bound variables.
- Let ϑ be admissible for r. Show that $r =_\alpha s$ implies $r\vartheta =_\alpha s\vartheta$ where $r\vartheta$ and $s\vartheta$ are arbitrary applications.

Definition (Alpha-equality of object terms)

Let r, s be object terms and let $((x_1\, y_1), \ldots, (x_n\, y_n))$ be a list of pairs of object variables. We say that r is equal-via-$((x_1\, y_1), \ldots, (x_n\, y_n))$ to s, if one of the following cases holds:

1. $r = x$, $s = y$ for object variables x, y and either
   - $x = y$, x not one of the x_i and y is not one of the y_i, or
   - there is a $j \in \{1, \ldots, n\}$ such that $x = x_j$, $y = y_j$ and $x \neq x_k$, $y \neq y_k$ for all $k \in \{j+1, \ldots, n\}$.
2. $r = \lambda_x r'$, $s = \lambda_y s'$, $\overline{x} = \overline{y}$ and r' is equal-via-$((x_1\, y_1), \ldots, (x_n\, y_n), (x\, y))$ to s'.
3. $r = r_1 r_2$, $s = s_1 s_2$ and $r_1 \;/\; r_2$ is equal-via-$((x_1\, y_1), \ldots, (x_n\, y_n))$ to $s_1 \;/\; s_2$.

Finally we say that r is alpha-equal to s, written $r =_\alpha s$, if r is equal-via-() to s.

Theorem

The relation $=_\alpha$ between object terms is an equivalence relation.

Proof: Requires some work, about three quarters of a A4 page. □

Theorem

Let r be an object term an let ϑ be a substitution admissible for r. If r_1 and r_2 are two applications of ϑ to r then we have $r_1 =_\alpha r_2$.

Proof: This is the case $n = 0$ in the following lemma. □

Lemma

Let ϑ be a substitution and r be an object term. Let $x_1, \ldots, x_n$, $y_1, \ldots, y_n$, $z_1, \ldots, z_n$ be object variables such that for all $i \in \{1, \ldots, n\}$ the following holds:

1. $\overline{x_i}\vartheta = \overline{y_i} = \overline{z_i}$.
2. *For all $z \in FV_o(r)$: If $y_i \in FV_o(z\vartheta^{y_1 \ldots y_{i-1}}_{x_1 \ldots x_{i-1}})$ then $z \in \{x_i, \ldots, x_n\}$.*
3. *For all $z \in FV_o(r)$: If $z_i \in FV_o(z\vartheta^{z_1 \ldots z_{i-1}}_{x_1 \ldots x_{i-1}})$ then $z \in \{x_i, \ldots, x_n\}$.*

Suppose that $\vartheta^{y_1 \ldots y_n}_{x_1 \ldots x_n}$ is admissible for r (hence so is $\vartheta^{z_1 \ldots z_n}_{x_1 \ldots x_n}$). Let r_1 / r_2 be two applications of $\vartheta^{y_1 \ldots y_n}_{x_1 \ldots x_n}$ / $\vartheta^{z_1 \ldots z_n}_{x_1 \ldots x_n}$ to r. Then r_1 is equal-via-$((y_1\, z_1), \ldots, (y_n\, z_n))$ to r_2.

Proof: Induction on r.
Case 1. $r = x$ for an object variable x. *Subcase 1.1.* x is not one of the x_i. Then we have $r_1 = x\vartheta = r_2$. We want to show that $x\vartheta$ is equal-via-$((y_1\, z_1), \ldots, (y_n\, z_n))$ to $x\vartheta$. This follows if we can show that $y_i, z_i \notin \mathrm{FV_o}(x\vartheta)$ for all $i \in \{1, \ldots, n\}$ (requires proof!). So show the latter: Suppose there exists $i \in \{1, \ldots, n\}$ such that $y_i \in \mathrm{FV_o}(x\vartheta)$. Since $x \notin \{x_1, \ldots, x_n\}$ in this subcase we have $x\vartheta^{y_1 \ldots y_{i-1}}_{x_1 \ldots x_{i-1}} = x\vartheta$. Because of $x \in \mathrm{FV_o}(r)$ it follows from (ii) that $x \in \{x_i, \ldots, x_n\}$ must hold. But the latter contradicts the assumption that $x \notin \{x_1, \ldots, x_n\}$. Analogously one uses (iii) to show that $z_i \notin \mathrm{FV_o}(x\vartheta)$ for all $i \in \{1, \ldots, n\}$. *Subcase 1.2.* $x = x_j$ and $x \neq x_k$ for $k > j$. Then $x\vartheta^{y_1 \ldots y_n}_{x_1 \ldots x_n} = y_j$ and $x\vartheta^{z_1 \ldots z_n}_{x_1 \ldots x_n} = z_j$. We need to show that for all $k \in \{j+1, \ldots, n\}$ we have $y_k \neq y_j$ and $z_k \neq z_j$. This can be done using (ii) and (iii) very similarly to subcase 1.1.

Case 2. $r = \lambda_x r'$. Then $r_1 = \lambda_{y_{n+1}} r' \vartheta^{y_1 \dots y_n y_{n+1}}_{x_1 \dots x_n x}$. Here y_{n+1} is an object variable satisfying $\overline{y_{n+1}} = \overline{x}\vartheta$ and $y_{n+1} \notin \bigcup_{z \in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta^{y_1 \dots y_n}_{x_1 \dots x_n})$. Furthermore $r' \vartheta^{y_1 \dots y_n y_{n+1}}_{x_1 \dots x_n x}$ is an application of $\vartheta^{y_1 \dots y_n y_{n+1}}_{x_1 \dots x_n x}$ to r'. Analogously $r_2 = \lambda_{z_{n+1}} r' \vartheta^{z_1 \dots z_n z_{n+1}}_{x_1 \dots x_n x}$. We would like to apply the induction hypothesis to $r' \vartheta^{y_1 \dots y_n y_{n+1}}_{x_1 \dots x_n x}$ and $r' \vartheta^{z_1 \dots z_n z_{n+1}}_{x_1 \dots x_n x}$ and therefore need to show that all conditions are satisfied. Let $i \in \{1, \dots, n+1\}$ be arbitrary. Obviously (i) is satisfied. Now consider (ii): Let $z \in \mathrm{FV_o}(r') \subseteq \mathrm{FV_o}(r) \cup \{x\}$. *Subcase 2.1.* If $z \in \mathrm{FV_o}(r)$ then the implication in (ii) is satisfied by assumption (for $i \in \{1, \dots, n\}$) and because of the variable condition $y_{n+1} \notin \bigcup_{z \in \mathrm{FV_o}(r)} \mathrm{FV_o}(z\vartheta^{y_1 \dots y_n}_{x_1 \dots x_n})$ (for $i = n+1$). *Subcase 2.2.* If $z = x$ then the implication in (ii) is satisfied because its consequent $z \in \{x_i, \dots, x_n, x\}$ is satisfied. Analogously one shows that (iii) is satisfied. Furthermore $\vartheta^{y_1 \dots y_n y_{n+1}}_{x_1 \dots x_n x}$ is admissible for r', as checked easily. Thus by induction hypothesis $r' \vartheta^{y_1 \dots y_n y_{n+1}}_{x_1 \dots x_n x}$ is

equal-via-$((y_1\, z_1), \ldots, (y_{n+1}\, z_{n+1}))$ to $r'\vartheta^{z_1 \ldots z_n z_{n+1}}_{x_1 \ldots x_n x}$. Using $\overline{y_{n+1}} = \overline{z_{n+1}}$ we may conclude that r_1 is equal-via-$((y_1\, z_1), \ldots, (y_n\, z_n))$ to r_2, as required.

Case 3. Easily shown using the induction hypothesis. Note that for $r = s_1 s_2$ we have $\mathrm{FV_o}(s_1), \mathrm{FV_o}(s_2) \subseteq \mathrm{FV_o}(r)$. Thus (ii) and (iii) hold for s_1 and s_2. □

Using this it is easy to prove the following:

Theorem

Let $r =_\alpha s$ be object terms and let ϑ be admissible for r. Then ϑ is admissible for s and we have $r\vartheta =_\alpha s\vartheta$ (for arbitrary applications).

The theory of substitutions yields an elegant characterization of alpha-equality:

Definition

A relation $\mathcal{R}$ on object terms is called *compatible with substitution* if the following holds: Let ϑ be admissible for an object term r and let r_1, r_2 be two applications of ϑ to r. Then $r_1 \mathcal{R} r_2$.

Theorem

Alpha-equality is the smalles relation on object terms which is compatible with substitution, that is: If $\mathcal{R}$ is a relation on object terms compatible with substitution and if r_1, r_2 are object terms then $r_1 =_\alpha r_2$ implies $r_1 \mathcal{R} r_2$.

- Goal: Define composition of substitutions such that $r(\eta \circ \vartheta) =_\alpha r\vartheta\eta$.
- Idea: Define the composition such that for each (type or object) variable X one has $X(\eta \circ \vartheta) =_\alpha X\vartheta\eta$.
- First Problem: $X\vartheta\eta$ is only defined modulo alpha-equality. Thus we will have to introduce a notion of alpha-equality of substitutions. (Not carried out in this talk.)
- Second Problem: For $X\vartheta\eta$ to be defined η needs to be admissible for $X\vartheta$. We will try to handle this problem in the following.

- Recall: Want to define $X(\eta \circ \vartheta) := X\vartheta\eta$. This requires that η is admissible for $X\vartheta$.
- Possible solution: Define $\eta \circ \vartheta$ only if η is admissible for $X\vartheta$ *for all* $X \in \mathrm{dom}(\vartheta)$.
- But this seems too restrictive, as the following example shows: Let r be an object term such that ϑ is admissible for r and η is admissible for $x\vartheta$ whenever $x \in \mathrm{FV_o}(r)$. On the other hand let η *not* be admissible for an object term $x\vartheta$ where $x \in \mathrm{dom}(\vartheta) \backslash \mathrm{FV_o}(r)$. We would then like to restrict ϑ to ϑ^* coinciding with ϑ on $\mathrm{FV_o}(r)$. Then we can define $\eta \circ \vartheta^*$ and get (as will be shown) $r(\eta \circ \vartheta^*) =_\alpha (r\vartheta)\eta$.
- Goal: Find a general strategy to carry out the described restrictions.

Definition

Let $\vartheta = ((X_1\ Y_1), \ldots, (X_n\ Y_n))$ and η be substitutions. The restriction ϑ_η of ϑ to η arises from $((X_1\ Y_1), \ldots, (X_n\ Y_n))$ by deleting all pairs $(X_i\ Y_i)$ where X_i is an object variable and η is not admissible for Y_i.

Definition

Let $\vartheta_\eta = ((X_1\ U_1), \ldots, (X_n\ U_n))$ and $\eta = ((Y_1\ V_1), \ldots, (Y_m\ V_m))$ be substitutions. For $i \in \{1, \ldots, n\}$ let $U_i\eta$ be an application of η to U_i. Consider the expression

$$((X_1\ U_1\eta), \ldots, (X_n\ U_n\eta), (Y_1\ V_1), \ldots, (Y_m\ V_m))$$

and delete all pairs $(X_i\ U_i\eta)$ where $X_i = U_i\eta$ and all pairs $(Y_i\ V_i)$ where $Y_i \in \{X_1, \ldots, X_m\}$. The result is called a composition of η after ϑ.

With this definition of composition the following holds:

Theorem

Let r be an object term and let ϑ and η be substitutions. If ϑ is admissible for r and η is admissible for $r\vartheta$ then $\eta \circ \vartheta$ is admissible for r and we have $r(\eta \circ \vartheta) =_\alpha r\vartheta\eta$.

- Price to pay for this solution: As a consequence of the restriction-operation composition is no longer assoziative. Consider the following:
  Let x, y be object variables and σ, τ be type variables such that $x \neq y$, $\overline{x} = \overline{y} = \sigma \neq \tau$. Let $\vartheta := ((x\ y))$, $\eta := ((\sigma\,\tau))$, $\xi := ((\tau\,\sigma))$. Because of $\overline{y\eta} = \overline{y} = \sigma \neq \tau = \overline{y}\eta$, η is not admissible for y. Thus $\vartheta_\eta = ()$ and $\eta \circ \vartheta = ((\sigma\,\tau))$. Then $\xi \circ (\eta \circ \vartheta) = ((\tau\,\sigma))$. On the other hand $\xi \circ \eta = ((\tau\,\sigma))$. Now $\overline{y(\xi \circ \eta)} = \overline{y} = \sigma = \overline{y}(\xi \circ \eta)$. Thus $\xi \circ \eta$ is admissible for y and thus $\vartheta_{\xi\circ\eta} = ((x\,y))$. Then $(\xi \circ \eta) \circ \vartheta = ((x\,y), (\tau\,\sigma))$. But clearly $((\tau\,\sigma)) \neq ((x\,y), (\tau\,\sigma))$.
- But if no restriction-operations are performed, then composition is associative, as can be shown.

- Formulas and comprehension terms: $Pt_1 \dots t_n \mid \pi := \{x_1, \dots, x_n | A\} \mid \pi t_1 \dots t_n \mid A \to B \mid \forall_x A$.
- For the moment we don't consider types of formulas (needed in the context of proof extraction). But predicates and comprehension terms do have arities.

Changes compared to object terms:

- Admissibility also for predicate variables.
- More then one variable abstracted at a time for comprehension term formation.
- No substantially new difficulties arise. All theorems for object terms can be transferred.

- Proof terms: $u^A \mid \lambda_u M \mid M^{A \to B} N^B \mid \lambda_x M \mid Mr$.
- The type of a proof term is the proved formula.

Changes compared to object terms and formulas:

- Admissibility for assumption variables must be defined as $\overline{u\vartheta} =_\alpha \overline{u}\vartheta$ since $\overline{u}\vartheta$ is only defined modulo alpha-equality.
- "Internal" substitution required for the spezialization in Mr. This makes it quite complicated to prove $\overline{M\vartheta} =_\alpha \overline{M}\vartheta$. My proof uses composition of substitutions in an essential way.
- Alpha-equality of proof terms must consider object *and* assumption variables.
- Because of free variables arising in different ways the proofs get much longer. Carefully dealing with the arising subtleties the theorems for object terms / formulas can be transferred.

Some simple cases:

- Assumption constants (axioms) built from uninstantiated formulas and internal substitutions.
- Predicates for arbitrary types, e.g. equality:

$$(\mathit{Eq}(\alpha)^{\alpha\to\alpha\to\mathbf{B}}; \alpha \mapsto \rho),$$

$$(\mathit{Eq}(\alpha)^{\alpha\to\alpha\to\mathbf{B}}; \alpha \mapsto \rho)\vartheta := (\mathit{Eq}(\alpha)^{\alpha\to\alpha\to\mathbf{B}}; \vartheta \circ (\alpha \mapsto \rho)).$$

More difficult cases:

- Recursion operator:
  - Is it appropriate to define $\mathcal{R}_\iota^\tau \vartheta := \mathcal{R}_{\iota\vartheta}^{\tau\vartheta}$?
  - Will this give $\overline{\mathcal{R}_\iota^\tau \vartheta} = \overline{\mathcal{R}_\iota^\tau} \vartheta$?
- Inductively defined predicates:
  - Similar questions as for the recurcion operator.
  - Will substitution commute with the formation of introduction and elimination axiom?
- Proof extraction: Under which assumptions will substitution commute with program extraction?

- Schwichtenberg, Helmut: *Minlog Reference Manual*. `http://minlog-system.de/`, accessed March 21, 2011.
- Schwichtenberg, Helmut: *Minimal Logic for Computable Functionals*. `http://minlog-system.de/`, accessed January 3, 2011.
- Schwichtenberg, Helmut: *Mathematical Logic*. `www.mathematik.uni-muenchen.de/chwicht/lehre.php`, accessed January 3, 2011.